The Comparison of Different Deep Learning Models for Facial Expression Recognition

DATS 6203 Machine Learning II
Group_1
Xiaochi Li, Liwei Zhu, Jia Chen
Instructor: Professor Amir Jafari
December 2018

Content

- Data Introduction
- Multilayer Perceptron
- Convolution neural network (CNN)
- Light VGG net: One CNN structure inspired by VGG
- Conclusions
- Application: Real time facial expression recognition

Data Introduction - Description

Real-world Affective Face Database (RAF-DB) Li, S., & Deng, W. (2018)

- 29,672 real-world facial images from Internet
- 2 subsets: single-label subset: 7 classes of basic emotions
  two-tabs subset: including 12 classes of compound emotions

single-label subset

- 15,339 images
- 7 class: “Surprised”, “Fearful”, “Disgusted”, “Happy”, “Sad”, “Angry”, and “Neural”.
- 12,271 training / 3,068 testing images
- crowd-sourcing
- 315 annotators label images into one of 7 basic emotions, and each image is annotated around 40 times.

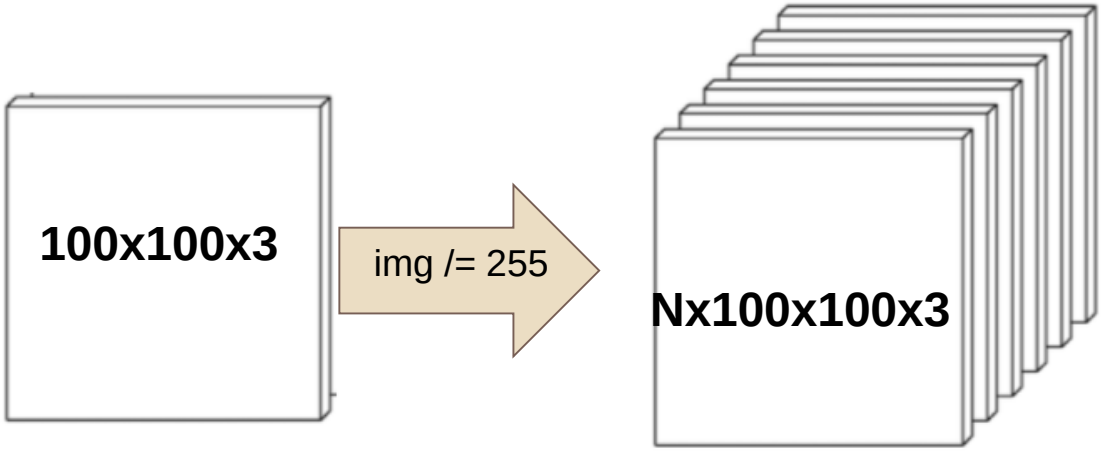
Data Introduction - data preprocessing

Create label dictionary

Data transformation and normalization

Save train/validation data as pickle

Dict : file name -> label



Training: 12271x100x100x3, 12271x7

Validation: 3068x100x100x3, 3068x7

Multilayer Perceptron

MLP - Initial setup

2-layer Perceptron

- input_dim = 100*100*3
- hidden1_num = 50
- output_num = 7
- optimizer = Adam
- learning rate = 0.001
- batch size = 32
- num_epoch = 5

Accuracy = 22.1%

```
def m_1():
    model = tf.keras.Sequential([
        layers.Dense(hidden1_num, activation='relu', input_dim=30000),
        layers.Dense(output_num, activation='softmax'),
    ])

    model.compile(optimizer=tf.train.AdamOptimizer(0.001),
                  loss='categorical_crossentropy',
                  metrics=[tf.keras.metrics.categorical_accuracy])

    return model
```

MLP - Parameters

Optimizer

Optimizer	Adam	Adadelta	RMSProp	SGD
Accuracy	22.1%	38.3%	38.5%	55.2%

Table 1. Optimizer

→ SGD is best!

Epochs

lr	Dropout	10	20	30	40	50
0.001	0.2	0.5968	0.6196	0.6231	0.6349	0.6643

Table 2. Accuracy for different epochs_lr_0.001

- Add dropout = 0.2
- Add num_epoch

Accuracy = 66.4%

lr

lr	Dropout	10	20	30	40	50
0.01	0.2	0.5209	0.528			0.6379

Table 3. Accuracy for different epochs_lr_0.01

- Add lr = 0.01

Accuracy = 63.8%

MLP - Parameter

Batch size

Batch size	32	64	128
Accuracy	0.6780	0.6670	0.6385
Time	123	132	137

Table 4. Accuracy for different batch size, lr_0.01

32 is best.
No big difference between 32 and 64.

Layer

Hidden layer					Output layer	Accuracy
L_1: 50					7	0.5968
L_1: 50	L_2: 10				7	0.3862
L_1: 50	L_2: 10	L_3: 10			7	0.3862
L_1: 10	L_2: 10	L_3: 10	L_4: 10	L_5: 10	7	0.3862

Table 5. Accuracy for different number of layer, epoch=10

- Adding layers

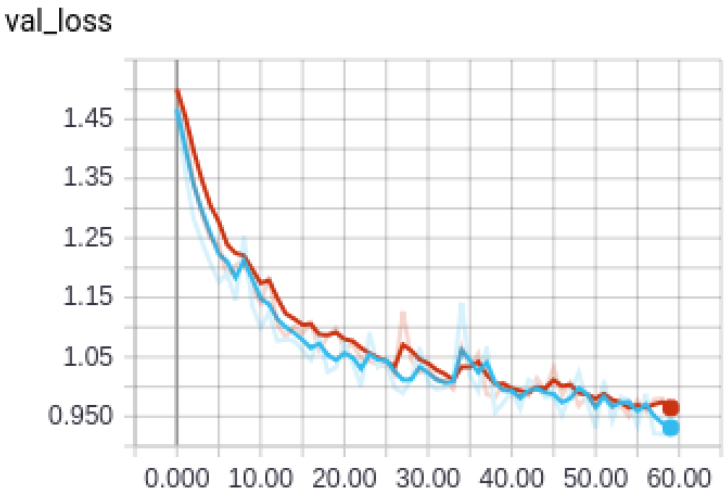
More layers decrease accuracy

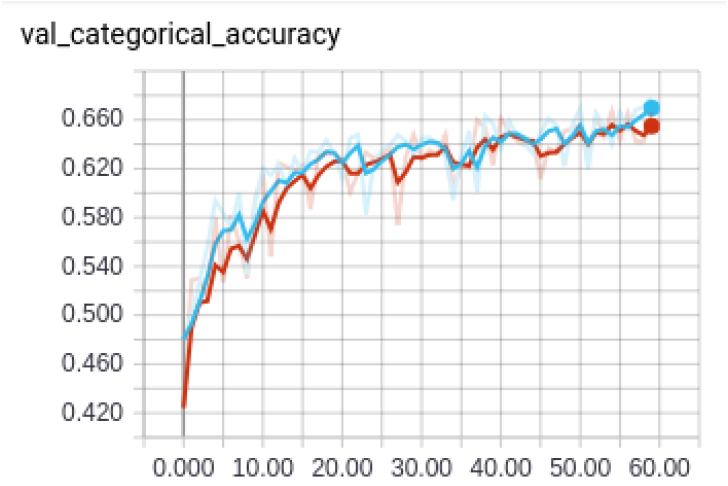
MLP - Result

Relatively good model

Model	Batch_size	Num_epoch	layer	Num_hidden	Num_output	Dropout	lr	Optimizer	Accuracy
m_3	64	60	2	50	7	0.2	0.001	SGD	66.7%
	32	60	2	50	7	0.2	0.001	SGD	67.8%

Table 6. Ideal models





Convolution Neural network (CNN)

CNN - Model structure

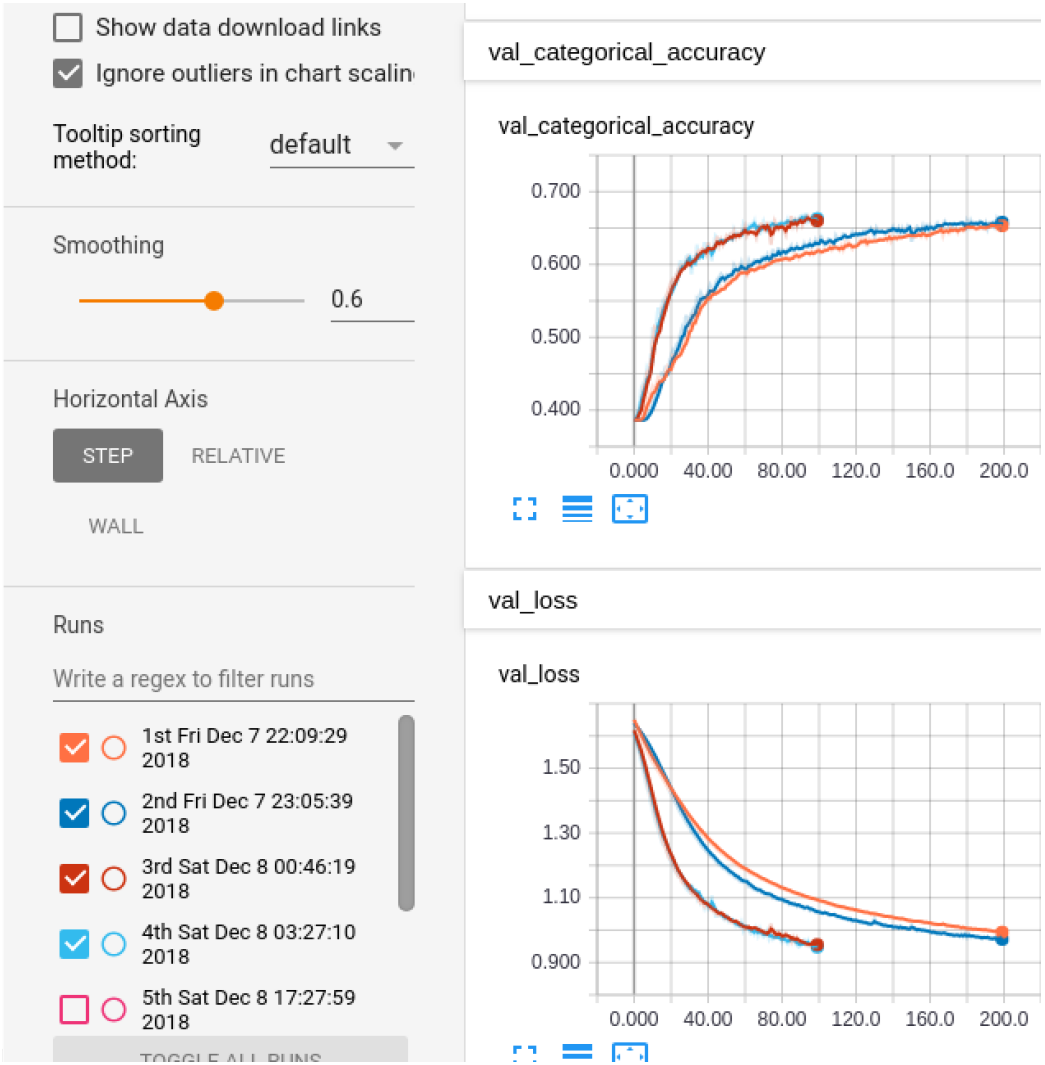
```
def first_model():
    model = tf.keras.Sequential([
        layers.Conv2D(64, kernel_size=3, activation='relu', input_shape=(100,100,3)),
        layers.Conv2D(32, kernel_size=3, activation='relu'),
        layers.Flatten(),
        layers.Dense(7, activation='softmax')
    ])
```

Added maxpooling layer

Modified the filter size

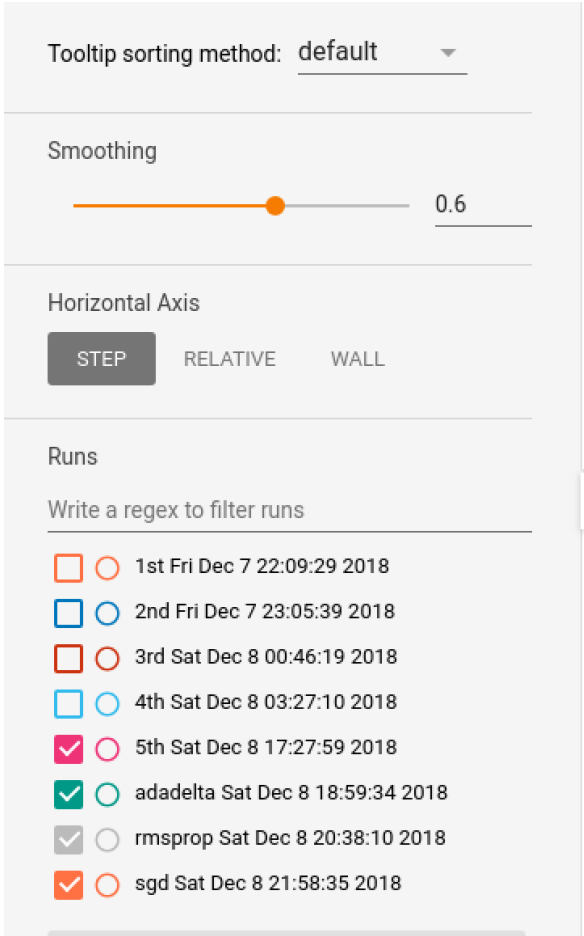
Added dropout and
fully connected layer

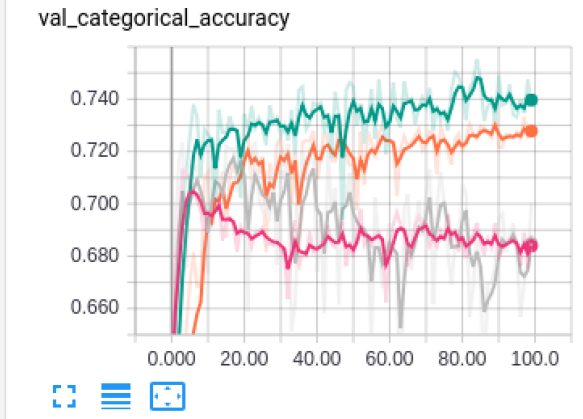
CNN - Model structure

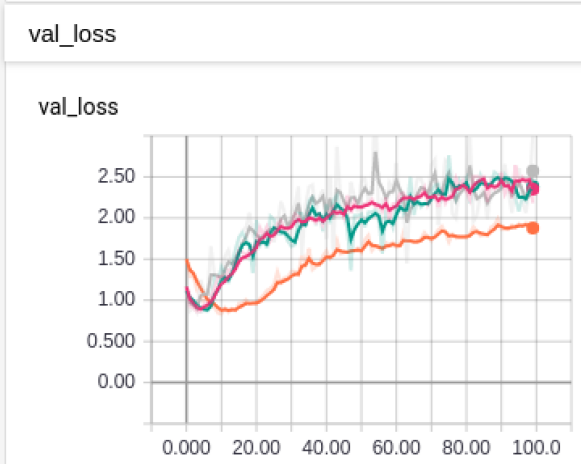


The validation accuracy of 4th model reached 66.62%

CNN - Optimizer



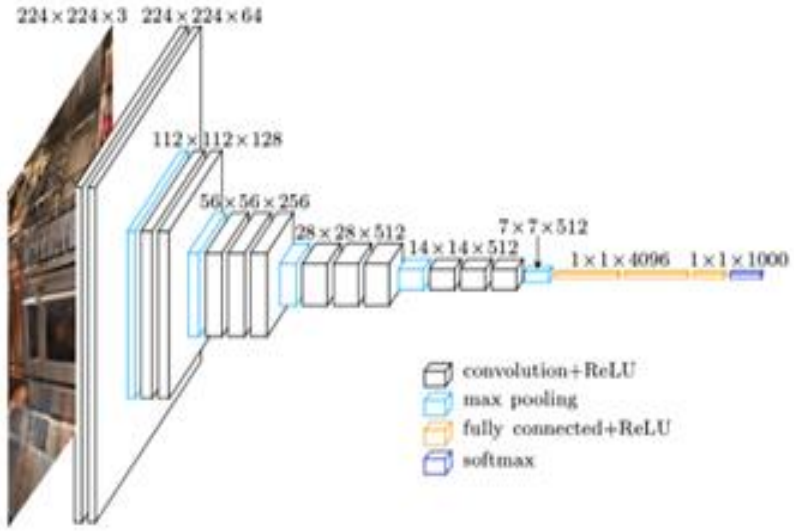


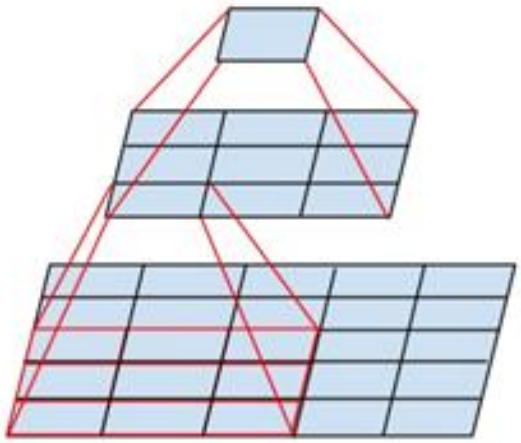


Adadelta optimizer performanced the highest validation accuracy above 74%

Light VGG net

VGG Net





Winner of ImageNet Large Scale Visual Recognition Challenge(2014)

By:University of Oxford

Multiple 3x3 convolutions in sequence

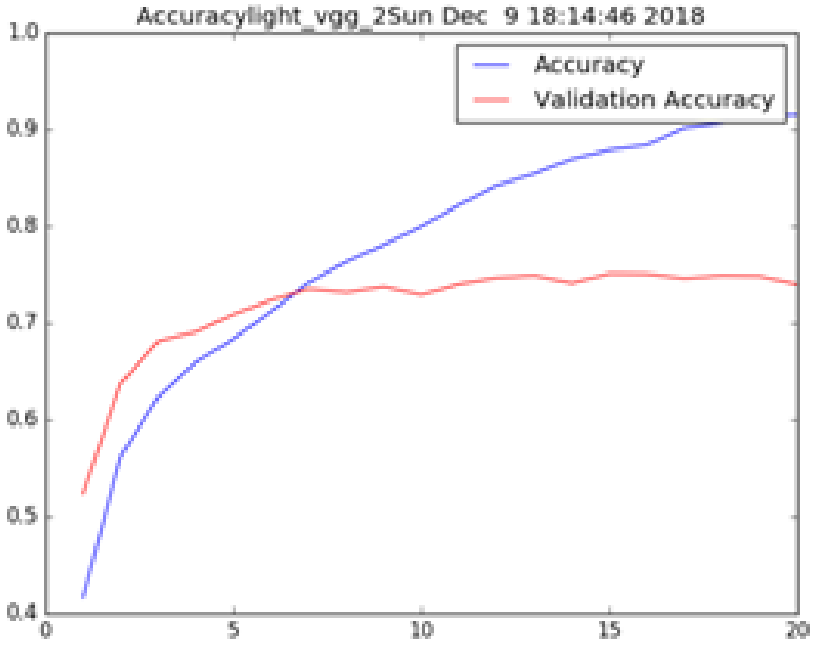
Original Problem: 224x224x3,1000

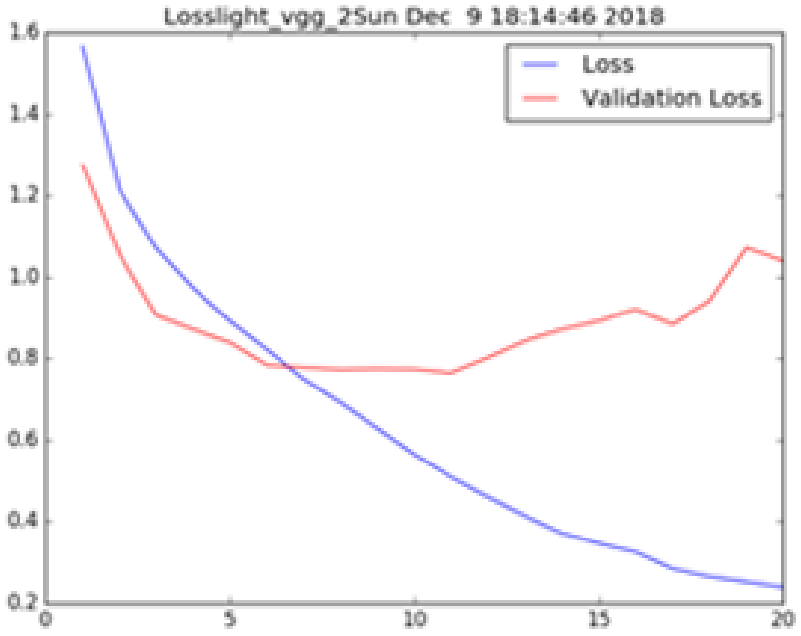
Our Problem: 100x100x3, 7

Simplify!

Light VGG V2

Conv2D (8 x (3x3)) Conv2D (8 x (3x3)) MaxPooling2D(2x2)	In 100x100x3 Out 50x50x8
Conv2D (16 x (3x3)) Conv2D (16 x (3x3)) MaxPooling2D(2x2)	In 50x50x8 Out 25x25x16
Dense (512) Drop (70%) Dense (512) Drop (70%)	In 10000 ~ 20:1 Out 512
Dense (7) Softmax	In 512 Out 7





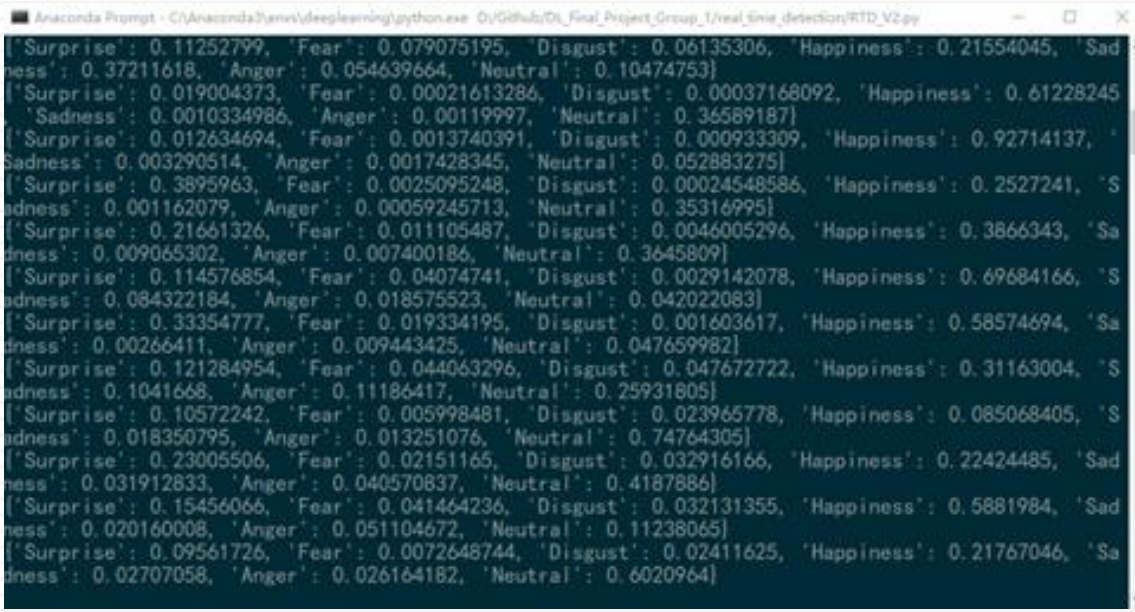
6s/epoch 11MB

Conclusions

- MLP is not a good model for expression classification on RAF-DB, as the accuracy is only 67.8%.
- CNN and light VGG did not generate a perfect model. The accuracy is around 74%, and both of the models has some overfitting issue.
- Light VGG V2 is our best model since it's faster and smaller and more suitable for real time facial expression recognition.
- Our future step could be collecting more data, pretrain the model on other dataset(eg. ImageNet) and use transfer learning.

Thank you!

Let’s see some real demo!



Back up

CNN - Adadelta

```
# Modified with Adadelta optimizer
# Accuracy around 99%/74% (epoch=100)
def Adadelta_model():
    model = tf.keras.Sequential([
        layers.Conv2D(128, kernel_size=3, activation='relu', input_shape=(100,100,3)),
        layers.Conv2D(128, kernel_size=3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.1),
        layers.Flatten(),
        layers.Dense(32, activation='relu'),
        layers.Dropout(0.1),
        layers.Dense(7, activation='softmax')
    ])

    model.compile(optimizer=tf.keras.optimizers.Adadelta(lr=1.0),
                  loss='categorical_crossentropy',
                metrics=[tf.keras.metrics.categorical_accuracy])

    return model
```

CNN - Adadelta